

[illegible]

```

UUU      UUU  EEEEEEEEEEEEEEEE  TTTTTTTTTTTTTTTT  PPPPPPPPPPPP  SSSSSSSSSSSS  YYY      YYY
UUU      UUU  EEEEEEEEEEEEEEEE  TTTTTTTTTTTTTTTT  PPPPPPPPPPPP  SSSSSSSSSSSS  YYY      YYY
UUU      UUU  EEEEEEEEEEEEEEEE  TTTTTTTTTTTTTTTT  PPPPPPPPPPPP  SSSSSSSSSSSS  YYY      YYY
UUU      UUU  EEE                TTT      PPP      PPP  SSS      YYY      YYY
UUU      UUU  EEE                TTT      PPP      PPP  SSS      YYY      YYY
UUU      UUU  EEE                TTT      PPP      PPP  SSS      YYY      YYY
UUU      UUU  EEE                TTT      PPP      PPP  SSS      YYY      YYY
UUU      UUU  EEE                TTT      PPP      PPP  SSS      YYY      YYY
UUU      UUU  EEE                TTT      PPP      PPP  SSS      YYY      YYY
UUU      UUU  EEE                TTT      PPP      PPP  SSS      YYY      YYY
UUU      UUU  EEE                TTT      PPP      PPP  SSS      YYY      YYY
UUU      UUU  EEE                TTT      PPP      PPP  SSS      YYY      YYY
UUU      UUU  EEEEEEEEEEEEEEEE  TTT      PPPPPPPPPPPP  SSSSSSSSSS  YYY
UUU      UUU  EEEEEEEEEEEEEEEE  TTT      PPPPPPPPPPPP  SSSSSSSSSS  YYY
UUU      UUU  EEEEEEEEEEEEEEEE  TTT      PPPPPPPPPPPP  SSSSSSSSSS  YYY
UUU      UUU  EEE                TTT      PPP      SSS      YYY
UUU      UUU  EEE                TTT      PPP      SSS      YYY
UUU      UUU  EEE                TTT      PPP      SSS      YYY
UUU      UUU  EEE                TTT      PPP      SSS      YYY
UUU      UUU  EEE                TTT      PPP      SSS      YYY
UUU      UUU  EEE                TTT      PPP      SSS      YYY
UUU      UUU  EEE                TTT      PPP      SSS      YYY
UUUUUUUUUUUUUUUUUU  EEEEEEEEEEEEEEEE  TTT      PPP      SSSSSSSSSSSS  YYY
UUUUUUUUUUUUUUUUUU  EEEEEEEEEEEEEEEE  TTT      PPP      SSSSSSSSSSSS  YYY
UUUUUUUUUUUUUUUUUU  EEEEEEEEEEEEEEEE  TTT      PPP      SSSSSSSSSSSS  YYY

```







(1)	54	DECLARATIONS
(1)	77	MACROS
(1)	129	R/W PSECT
(1)	210	SATSSS43
(1)	261	DCLCMH TESTS
(1)	358	DCLEXH TESTS #1
(1)	412	CANEXH TESTS
(1)	458	DCLEXH TESTS #2
(2)	522	SUPER MODE
(2)	587	USER MODE
(2)	616	COMP MODE
(2)	648	SETUP SUPER ROUTINE
(2)	728	REG_SAVE
(2)	750	REG_CHECK
(2)	793	REG_CHECKNP
(2)	857	ERLBUF DUMP
(2)	895	PRINT_FAIL
(2)	952	MODE_ID



```
0000 1 .TITLE SATSSS43 - SATS SYSTEM SERVICE TESTS (SUCC S.C.)
0000 2 .IDENT 'V04-000'
0000 3
0000 4
0000 5 *****
0000 6
0000 7 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 8 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 9 * ALL RIGHTS RESERVED.
0000 10
0000 11 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 12 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 13 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 14 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 15 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 16 * TRANSFERRED.
0000 17
0000 18 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 19 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 20 * CORPORATION.
0000 21
0000 22 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 23 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 24
0000 25 *
0000 26 *****
0000 27
0000 28
0000 29 ++
0000 30 FACILITY: SATS SYSTEM SERVICE TESTS
0000 31
0000 32 ABSTRACT: The SATSSS43 module tests the execution of the following
0000 33 VMS system services:
0000 34
0000 35 $DCLCMH
0000 36 $DCLEXH
0000 37 $SCANEXH
0000 38
0000 39
0000 40 ENVIRONMENT: User, Supervisor and Executive mode image.
0000 41 Needs CMKRNL privilege and dynamically acquires other
0000 42 privileges, as needed.
0000 43
0000 44 AUTHOR: THOMAS L. CAFARELLA, CREATION DATE: MMM, 1978
0000 45 PAUL D. FAY (DISPSERV & TESTSERV MACROS)
0000 46
0000 47 MODIFIED BY:
0000 48
0000 49 V03-001 LDJ0001 Larry D. Jones, 17-Sep-1980
0000 50 Modified to conform to new build command procedures.
0000 51 **
0000 52 --
```



```
0000 54 .SBTTL DECLARATIONS
0000 55 :
0000 56 : MACRO LIBRARY CALLS
0000 57 :
0000 58 $PCBDEF : PCB definitions
0000 59 $PHDDEF : process header definitions
0000 60 $PRDEF : processor register definitions
0000 61 $PRVDEF : privilege definitions
0000 62 $PSLDEF : PSL definitions
0000 63 $SFDEF : Stack Frame definitions
0000 64 $SHR MESSAGES UETP,116,<<TEXT,INFO>> : UETPS_TEXT definition
0000 65 $SSDEF : system service definitions
0000 66 $STSDEF : STS definitions
0000 67 $UETPDEF : UETP message definitions
0000 68 :
0000 69 : Equated symbols
0000 70 :
00000000 0000 71 WARNING = 0 : warning severity value for msgs
00000001 0000 72 SUCCESS = 1 : success
00000002 0000 73 ERROR = 2 : error
00000003 0000 74 INFO = 3 : information
00000004 0000 75 SEVERE = 4 : fatal
0000 76 :
0000 77 .SBTTL MACROS
0000 78 :
0000 79 .MACRO EHDB MODE,NUM
0000 80 .LIST MEB
0000 81 MODE'NUM:
0000 82 .LONG 0
0000 83 .ADDRESS MODE'H'NUM
0000 84 .LONG 2
0000 85 .ADDRESS STATUS
0000 86 .LONG NUM
0000 87 .NLIST MEB
0000 88 .ENDM EHDB
0000 89 :
```



SATSSS43  
V04-000

- SATS SYSTEM SERVICE TESTS (SUCC S.C.)  
MACROS

N 5

16-SEP-1984 00:54:19 VAX/VMS Macro V04-00  
5-SEP-1984 04:31:29 [UETPSY.SRC]SATSSS43.MAR;1

Page 3  
(1)

```
00000000 91 .PSECT RODATA, RD, NOWRT, NOEXE, LONG
0000 92
0000 93 TEST_MOD_NAME:
33 34 53 53 53 54 41 53 00' 0000 94 .ASCIC /SATSSS43/ ; needed for SATSMS message
08 0000
0009 95 TEST_MOD_NAME D:
53 53 53 54 41 53 00000011'010E0000' 0009 96 .ASCID /SATSSS43/ ; module name
33 34 0017
0019 97 TEST_MOD_BEGIN:
6E 75 67 65 62 00' 0019 98 .ASCIC /begun/
05 0019
001F 99 TEST_MOD_SUCC:
6C 75 66 73 73 65 63 63 75 73 00' 001F 100 .ASCIC /successful/
0A 001F
002A 101 TEST_MOD_FAIL:
64 65 6C 69 61 66 00' 002A 102 .ASCIC /failed/
06 002A
0031 103 DCCLCMH:
48 4D 43 4C 43 44 00' 0031 104 .ASCIC /DCCLCMH/
06 0031
0038 105 DCLEXH:
48 58 45 4C 43 44 00' 0038 106 .ASCIC /DCLEXH/
06 0038
003F 107 CANEXH:
48 58 45 4E 41 43 00' 003F 108 .ASCIC /CANEXH/
06 003F
0046 109 CS1:
21 20 74 73 65 54 0000004E'010E0000' 0046 110 .ASCID \Test !AC service name !AC step !UL failed.\
6E 20 65 63 69 76 72 65 73 20 43 41 0054
70 65 74 73 20 43 41 21 20 65 6D 61 0060
2E 64 65 6C 69 61 66 20 4C 55 21 20 006C
0078 111 CS2:
74 63 65 70 78 45 00000080'010E0000' 0078 112 .ASCID \Expected !AS = !XL received !AS = !XL\
4C 58 21 20 3D 20 53 41 21 20 64 65 0086
41 21 20 64 65 76 69 65 63 65 72 20 0092
4C 58 21 20 3D 20 53 009E
00A5 113 CS3:
74 63 65 70 78 45 000000AD'010E0000' 00A5 114 .ASCID \Expected !AS!UB = !XL received !AS!UB = !XL\
20 3D 20 42 55 21 53 41 21 20 64 65 00B3
64 65 76 69 65 63 65 72 20 4C 58 21 00BF
58 21 20 3D 20 42 55 21 53 41 21 20 00CB
4C 00D7
00D8 115 CS4:
65 70 78 65 6E 55 000000E0'010E0000' 00D8 116 .ASCID \Unexpected !AS mode exit handler found in !AS.\
64 6F 6D 20 53 41 21 20 64 65 74 63 00E6
6C 64 6E 61 68 20 74 69 78 65 20 65 00F2
20 6E 69 20 64 6E 75 6F 66 20 72 65 00FE
2E 53 41 21 010A
010E 117 CS5:
77 20 65 64 6F 4D 00000116'010E0000' 010E 118 .ASCID \Mode was !AS.\
2E 53 41 21 20 73 61 011C
0123 119 UM:
72 65 73 75 0000012B'010E0000' 0123 120 .ASCID \user\
012F 121 SM:
72 65 70 75 73 00000137'010E0000' 012F 122 .ASCID \super\
013C 123 EM:
74 75 63 65 78 65 00000144'010E0000' 013C 124 .ASCID \executive\
```

SAT  
V04



SATSSS43  
V04-000

- SATS SYSTEM SERVICE TESTS (SUCC S.C.)<sup>B 6</sup> 16-SEP-1984 00:54:19 VAX/VMS Macro V04-00  
MACROS 5-SEP-1984 04:31:29 [UETPSY.SRC]SATSSS43.MAR;1

Page 4  
(1)

65 76 69 014A

014D

73 75 74 61 74 73 00000155'010E0000' 014D 125 EXP:

126

.ASCID \status\



```
015B 128 ;
015B 129 ;
00000000 130 .SBTTL R/W PSECT
0000 131 .PSECT RWDATA,RD,WRT,NOEXE,LONG
0000 132 ;
00000000 133 IPID: .LONG 0 ; PID for this process
0000 134 CURRENT_TC: .LONG 0 ; ptr to current test case
00000000 135 .ALIGN LONG
0008 136 REG_SAVE_AREA: .BLKL 15 ; register save area
00000044 137 MOD_MSG_CODE: .LONG UETPS_SATSMS ; test module message code for putmsg
007480D9 138 TMN_ADDR: .ADDRESS TEST_MOD_NAME
00000000 139 TMD_ADDR: .ADDRESS TEST_MOD_BEGIN
00000019 140 PRVPRT: .BYTE 0 ; protection return byte for SETPRT
00 0050 141 PRIVMASK: .QUAD 0 ; priv. mask
00000000 00000000 0051 142 CHM_CONT: .LONG 0 ; change mode continue address
00000000 0059 150 RETADR: .BLKL 2 ; returned address's from SETPRT
00000065 005D 151 STATUS: .LONG 0
00000000 0065 152 MODE: .LONG 0
00000000 0069 153 DCL: $DCLCMH DUMMY,OHC,0 ; DCLCMH parameter list
006D 154 DCL1: $DCLEXH EXEC3 ; DCLEXH parameter list
006D 155 CAN: $CANEXH EXEC1 ; CANEXH parameter list
007D 156 REG: .ASCID \register R\
007D 157 REGNUM: .LONG 0 ; register number
0085 158 MSGL: .LONG 80 ; buffer desc.
0085 159 BUF: .ADDRESS BUF
008D 160 BUF: .BLKB 80
009B 161 MESSAGEL: .LONG 0 ; message desc.
009F 162 SERV_NAME: .ADDRESS BUF
00A3 163 PRVHND1: .LONG 0 ; service name pointer
00A3 164 PRVHND2: .LONG 0 ; previous handler address 1
00A7 165 PRVHND3: .LONG 0 ; previous handler address 2
00AB 166 OHC: .LONG 0 ; previous handler address 3
00AB 167
00FB 168
00FB 169
00FF 170
0103 171
0103 172
0107 173
0107 174
010B 175
010B 176
010F 177
010F 178
0113 179
```



SATSSS43  
V04-000

- SATS SYSTEM SERVICE TESTS (SUCC S.C.) 16-SEP-1984 00:54:19 VAX/VMS Macro V04-00  
R/W PSECT 5-SEP-1984 04:31:29 [UETPSY.SRC]SATSSS43.MAR;1

Page 6  
(1)

00000000	0113	184	.LONG	0	; old handler check location
	0117	185	ARGLST:		
00000001	0117	186	.LONG	1	; super mode setup arg list
000003E6	011B	187	.ADDRESS	SUPER_MODE	
	011F	188	MSGVEC:		; PUTMSG message vector
00000003	011F	189	.LONG	3	
00741133	0123	190	.LONG	UETPS_TEXT	
00000001	0127	191	.LONG	1	
000000FB	012B	192	.ADDRESS	MESSAGEL	
	012F	193	MSGVEC1:		
00000004	012F	194	.LONG	4	; PUTMSG message vector for exit
00000000	0133	195	.LONG	0	
00000002	0137	196	.LONG	2	
00000143	013B	197	.BLKL	2	



```
0143 199 ; exit handler desc. blocks
0143 200 USER1: EHDB USER,1 ; user #1 will be deleted
0143
00000000 0143 .LONG 0
00000735 0147 .ADDRESS USERH1
00000002 014B .LONG 2
00000065 014F .ADDRESS STATUS
00000001 0153 .LONG 1
0157 201 USER2: EHDB USER,2 ; user #2 will be used
0157
00000000 0157 .LONG 0
00000353 015B .ADDRESS USERH2
00000002 015F .LONG 2
00000065 0163 .ADDRESS STATUS
00000002 0167 .LONG 2
016B 202 USER3: EHDB USER,3 ; user #3 will be deleted
016B
00000000 016B .LONG 0
00000735 016F .ADDRESS USERH3
00000002 0173 .LONG 2
00000065 0177 .ADDRESS STATUS
00000003 017B .LONG 3
017F 203 USER4: EHDB USER,4 ; user #4 will be used
017F
00000000 017F .LONG 0
0000034C 0183 .ADDRESS USERH4
00000002 0187 .LONG 2
00000065 018B .ADDRESS STATUS
00000004 018F .LONG 4
0193 204 SUPER1: EHDB SUPER,1 ; super #1 will be deleted
0193
00000000 0193 .LONG 0
00000740 0197 .ADDRESS SUPERH1
00000002 019B .LONG 2
00000065 019F .ADDRESS STATUS
00000001 01A3 .LONG 1
01A7 205 SUPER3: EHDB SUPER,3 ; super #3 will be deleted
01A7
00000000 01A7 .LONG 0
00000740 01AB .ADDRESS SUPERH3
00000002 01AF .LONG 2
00000065 01B3 .ADDRESS STATUS
00000003 01B7 .LONG 3
01BB 206 EXEC1: EHDB EXEC,1 ; exec #1 will be deleted
01BB
00000000 01BB .LONG 0
0000074B 01BF .ADDRESS EXECH1
00000002 01C3 .LONG 2
00000065 01C7 .ADDRESS STATUS
00000001 01CB .LONG 1
01CF 207 EXEC3: EHDB EXEC,3 ; exec #3 will be deleted
01CF
00000000 01CF .LONG 0
0000074B 01D3 .ADDRESS EXECH3
00000002 01D7 .LONG 2
00000065 01DB .ADDRESS STATUS
00000003 01DF .LONG 3
```



```
00000000 209      .PSECT SATSSS43,RD,WRT,EXE,LONG
0000      210      .SBTTL SATSSS43
0000      211      :++
0000      212      : FUNCTIONAL DESCRIPTION:
0000      213      :
0000      214      :     After performing some initial housekeeping, such as
0000      215      :     printing the module begin message and acquiring needed privileges,
0000      216      :     the system services are tested in each of their normal conditions.
0000      217      :     Detected failures are identified and an error message is printed
0000      218      :     on the terminal. Upon completion of the test a success or fail
0000      219      :     message is printed on the terminal.
0000      220      :
0000      221      : CALLING SEQUENCE:
0000      222      :
0000      223      :     $ RUN SATSSS43 ... (DCL COMMAND)
0000      224      :
0000      225      : INPUT PARAMETERS:
0000      226      :
0000      227      :     none
0000      228      :
0000      229      : IMPLICIT INPUTS:
0000      230      :
0000      231      :     none
0000      232      :
0000      233      : OUTPUT PARAMETERS:
0000      234      :
0000      235      :     none
0000      236      :
0000      237      : IMPLICIT OUTPUTS:
0000      238      :
0000      239      :     Messages to SYS$OUTPUT are the only output from SATSSS43.
0000      240      :     They are of the form:
0000      241      :
0000      242      :         XUETP-S-SATSMS, TEST MODULE SATSSS43 BEGUN ... (BEGIN MSG)
0000      243      :         XUETP-S-SATSMS, TEST MODULE SATSSS43 SUCCESSFUL ... (END MSG)
0000      244      :         XUETP-E-SATSMS, TEST MODULE SATSSS43 FAILED ... (END MSG)
0000      245      :         XUETP-I-TEXT, ... (VARIABLE INFORMATION ABOUT A TEST MODULE FAILURE)
0000      246      :
0000      247      : COMPLETION CODES:
0000      248      :
0000      249      :     The SATSSS43 routine terminates with a $EXIT to the
0000      250      :     operating system with a status code defined by UETP$_SATSMS.
0000      251      :
0000      252      : SIDE EFFECTS:
0000      253      :
0000      254      :     none
0000      255      :
0000      256      : --
0000      257      :
0000      258      :
0000      259      :
0000      260      : TEST_START SATSSS43                ; let the test begin
```



```
0000 0000
0004'CF 00 DD 0002
0000'CF DF 0006
00000000'GF 02 FB 000C
00000000'GF 00 FB 0013
0009'CF 7F 001A
00000000'GF 01 FB 001E
0798 30 0025
004C'CF 001F'CF DE 0028
0044'CF 03 00 01 FO 002F
00 DD 0036
0549'CF 01 FB 0038
003D
003D 261 STP0:
003D 262 .SBTTL DCLCMH TESTS
003D 263 :+
003D 264 : $DCLCMH tests
003D 265 : test super mode handler declaration
003D 266 :-
003D 267
0103'CF 0031'CF DE 003D 268 MOVAL W^DCLCMH,W^SERV_NAME ; set service name
0663'CF 00 FB 0044 269 $CMKRN S W^SETUP SUPER,W^ARGLST ; test super mode declaration
0113'CF 000003E6'8F 01 BE 0053 270 CALLS #0,W^ERLBOF_DUMP ; report any errors
0113'CF 11 13 0058 271 CHMS #1 ; declare dummy handler
0113'CF DD 0063 272 CMPL #SUPER_MODE,W^OHC ; make sure it happened
03E6'CF DF 0065 273 BEQL 10$ ; br if yes
014D'CF DF 0069 274 PUSHL W^OHC ; else setup to report the error
0691'CF 03 FB 006D 275 PUSHAL W^SUPER_MODE ; save the expected results
02 BE 0071 276 PUSHAL W^EXP ; push the message address
0076 277 CALLS #3,W^PRINT_FAIL ; report the failure
0076 278 10$:
0076 279 CHMS #2 ; remove the dummy handler
0078 280 :+
0078 281 : test user mode handler declaration
0078 282 :-
0078 283
0078 284
0078 285 NEXT_TEST
0078
0004'CF 01 DO 0078
0549'CF 00 DD 007D
0069'CF 0123'CF 01 FB 007F
0071'CF 049A'CF DE 0084 286 MOVAL W^UM,W^MODE ; set the mode
0075'CF 010B'CF DE 008B 287 MOVAL W^DUMMY,W^DCL+DCLCMH$_ADDRES ; reset the handler address
0099 288 MOVAL W^PRVHND2,W^DCL+DCLCMH$_PRVHND ; set new handler save address
00A2 289 $DCLCMH G W^DCL ; check G form
00A2 290 FAIL_CHECK $$$_NORMAL ; check for success
0553'CF 01 DD 00A2
0553'CF 01 FB 00A4
00A9 291 $DCLCMH S W^USER MODE,W^OHC ; set real handler
00BA 292 FAIL_CHECK $$$_NORMAL ; check for success
01 DD 00BA
0553'CF 01 FB 00BC
0113'CF 0000049A'8F D1 00C1 293 CMPL #DUMMY,W^OHC ; is handler address correct?
```



```

      0113'CF 11 13 00CA 294      BEQL 10$      ; br if yes
      049A'CF 01 DD 00CC 295      PUSHL W^OHC      ; push received address
      014D'CF 01 DF 00D0 296      PUSHAL W^DUMMY      ; push expected address
0691'CF 03 DF 00D4 297      PUSHAL W^EXP      ; push string variable
      FB 00D8 298      CALLS #3,W^PRINT_FAIL      ; print the error
      00DD 299 10$:
      00DD 300 :+
      00DD 301 :
      00DD 302 : test for compatibility mode handler declaration
      00DD 303 :
      00DD 304 :-
      00DD 305      NEXT_TEST
      00DD
0004'CF 02 DO 00DD      STP2:
      00 00E2
0549'CF 01 FB 00E4
0075'CF 010F'CF DE 00E9 306      MOVAL W^PRVHND3,W^DCL+DCLCMH$_PRVHND ; set new handler save location
      0079'CF D6 00F0 307      INCL W^DCL+DCLCMH$_TYPE      ; set to compatibility mode type
      00FD 308      $DCLCMH G W^DCL      ; check G form
      00FD 309      FAIL_CHECK $$$_NORMAL      ; check for success
      01 00FD
0553'CF 01 FB 00FF
      01 0104 310      $DCLCMH S W^COMP MODE,W^OHC,#1      ; set real handler
      0115 311      FAIL_CHECK $$$_NORMAL      ; check for success
      01 0115
0553'CF 01 FB 0117
0113'CF 0000049A'8F D1 011C 312      CMPL #DUMMY,W^OHC      ; is handler address correct?
      11 13 0125 313      BEQL 10$      ; br if yes
      0113'CF DD 0127 314      PUSHL W^OHC      ; push received address
      049A'CF DF 012B 315      PUSHAL W^DUMMY      ; push expected address
      014D'CF DF 012F 316      PUSHAL W^EXP      ; push string variable
0691'CF 03 FB 0133 317      CALLS #3,W^PRINT_FAIL      ; print the error
      0138 318 10$:
      0138 319 :+
      0138 320 :
      0138 321 : check the compatibility mode handler
      0138 322 :
      0138 323 :-
      0138 324      NEXT_TEST
      0138
0004'CF 03 DO 0138      STP3:
      00 DD 013D
0549'CF 01 FB 013F
83C00000 8F DD 0144 325      PUSHL #<<PSL$M CM>!<PSL$C_USER@PSL$V_PRVMOD>-
      4E'AF DF 014A 326      !<PSL$C_USER@PSL$V_CURMOD>>      ; set compatibility mode
      02 014D 327      PUSHAL B^10$      ; set new address
      REI 328      ; enter compatibility mode
```



```

014E 330 .ALIGN WORD ; adjust addressing for PDP-11's
014E 331 10$:
014E 332 .WORD ^0012767 ; MOV #-1,TEST ;prove we were here
FFFF 333 .WORD ^0177777
0002 334 .WORD ^0000002
0000 335 .WORD ^0000000 ; HALT ;cause an exception
0156 336 TEST:
0156 337 .WORD ^0000000 ; compatibility mode flag location
0158 338 RETURN: ; return to the good life
0158 339 :+
0158 340 :
0158 341 : test the user mode handler
0158 342 :
0158 343 :-
0158 344 NEXT_TEST
0158
0158 STP4:
0158 MOVL #4,W^CURRENT_TC
015D PUSHL #0
015F CALLS #1,W^REG_SAVE
0164 345 CHMU #5 ; use a param of 5
0166 346 :+
0166 347 :
0166 348 : reset handlers to the original address
0166 349 :
0166 350 :-
0166 351 NEXT_TEST
0166
0166 STP5:
0166 MOVL #5,W^CURRENT_TC
016B PUSHL #0
016D CALLS #1,W^REG_SAVE
0172 352 MOVAL W^DCLCMH,W^SERV_NAME ; set service name
0179 353 $DCLCMH S 0,W^PRVHND2 ; reset CHMU handler
0188 354 FAIL_CHECK $$$_NORMAL ; check for success
0188 PUSHL $$$_NORMAL
018A CALLS #1,W^REG_CHECK
018F 355 $DCLCMH S 0,W^PRVHND3,#1 ; reset CM handler
019E 356 FAIL_CHECK $$$_NORMAL ; check for success
019E PUSHL $$$_NORMAL
01A0 CALLS #1,W^REG_CHECK

```



```
01A5 358 .SBTTL DCLEXH TESTS #1
01A5 359 :+
01A5 360 :
01A5 361 : $DCLEXH tests
01A5 362 :
01A5 363 : These tests are divided into two parts. This part is the declaration
01A5 364 : tests. The second part is the servicing part.
01A5 365 :
01A5 366 : test for exec mode exit handler declaration
01A5 367 :
01A5 368 :-
01A5 369 NEXT_TEST
01A5
01A5 STP6:
0004'CF 06 DO 01A5 MOVL #6,W^CURRENT_TC
0000 00 DD 01AA PUSHL #0
0549'CF 01 FB 01AC CALLS #1,W^REG_SAVE
0069'CF 013C'CF DE 01B1 MOVAL W^EM,W^MODE ; set the mode
0103'CF 0038'CF DE 01B8 MOVAL W^DCLEXH,W^SERV_NAME ; set service name
2C 11 01BF $CMEXEC S B^10$ ; get to exec mode
01CD 373 BRB -20$ ; skip over exec routine
01CD 374 10$:
0000 01CD .WORD 0
0000 01CF PUSHL #0 ; push a dummy parameter
0549'CF 01 FB 01D1 CALLS #1,W^REG_SAVE ; save a reg snapshot
01D6 377 $DCLEXH S W^EXEC ; declare #1 exec exit handler
01E1 378 FAIL_CHECKNP SSS NORMAL ; check for success
01E1 379 PUSHL #SS$ NORMAL
05EA'CF 01 FB 01E3 CALLS #1,W^REG_CHECKNP
01E8 380 $DCLEXH G W^DCL1 ; declare #3 exec exit handler
01F1 381 FAIL_CHECKNP SSS NORMAL ; check for success
01F1 382 PUSHL #SS$ NORMAL
05EA'CF 01 FB 01F3 CALLS #1,W^REG_CHECKNP
04 01F8 382 RET ; go back to user mode
01F9 383 20$:
0663'CF 00 FB 01F9 384 CALLS #0,W^ERLBUF_DUMP ; dump any errors that occurred
01FE 385 :+
01FE 386 :
01FE 387 : test super mode exit handler declaration
01FE 388 :
01FE 389 :-
01FE 390 NEXT_TEST
01FE
01FE STP7:
0004'CF 07 DO 01FE MOVL #7,W^CURRENT_TC
0000 00 DD 0203 PUSHL #0
0549'CF 01 FB 0205 CALLS #1,W^REG_SAVE
0069'CF 012F'CF DE 020A MOVAL W^SM,W^MODE ; set the mode
04 BE 0211 391 CHMS #4 ; declare 2 super mode exit handlers
392
```



```
0213 394 :+
0213 395 :
0213 396 : test user mode exit handler declaration
0213 397 :
0213 398 :-
0213 399 :
                                NEXT_TEST1
0213 STP8:
0004'CF 08 DO 0213          MOVL #8,W^CURRENT_TC
                                PUSHL #0
0549'CF 01 DD 0218          CALLS #1,W^REG_SAVE
0069'CF 0123'CF DE 021A      MOVAL W^UM,W^MODE ; set the mode
                                SDCLEXH S W^USER1 ; declare #1 user mode exit handler
                                FAIL_CHECK SSS_NORMAL ; check for success
                                PUSHL #SS$ NORMAL
0553'CF 01 DD 0231          CALLS #1,W^REG_CHECK
0081'CF 0157'CF DE 0233      MOVAL W^USER2,W^DCL1+DCLEXH$_DESBLK ; set exit handler address
                                SDCLEXH G W^DCL1 ; declare #2 user mode exit handler
                                FAIL_CHECK SSS_NORMAL ; check for success
                                PUSHL #SS$ NORMAL
0553'CF 01 DD 0248          CALLS #1,W^REG_CHECK
                                SDCLEXH S W^USER3 ; declare #3 user mode exit handler
                                FAIL_CHECK SSS_NORMAL ; check for success
                                PUSHL #SS$ NORMAL
0553'CF 01 DD 025A          CALLS #1,W^REG_CHECK
0081'CF 017F'CF DE 024A      MOVAL W^USER4,W^DCL1+DCLEXH$_DESBLK ; set exit handler address
                                SDCLEXH G W^DCL1 ; declare #4 user mode exit handler
                                FAIL_CHECK SSS_NORMAL ; check for success
                                PUSHL #SS$ NORMAL
0553'CF 01 DD 0271          CALLS #1,W^REG_CHECK
                                FB 0273
```



```
0278 412 .SBTTL CANEXH TESTS
0278 413 :+
0278 414 :
0278 415 : $CANEXH tests
0278 416 : test for exec mode exit handler deletion
0278 417 :
0278 418 :-
0278 419 NEXT_TEST
0278
0278 STP9:
0004'CF 09 DO 0278 MOVL #9,W^CURRENT_TC
0000 00 DD 027D PUSHL #0
0549'CF 01 FB 027F CALLS #1,W^REG_SAVE
0069'CF 013C'CF DE 0284 420 MOVAL W^EM,W^MODE ; set the mode
0103'CF 003F'CF DE 028B 421 MOVAL W^CANEXH,W^SERV_NAME ; set service name
0292 422 $CMEXEC S B^10$ ; get to exec mode
029E 423 BRB -20$ ; skip over the routine
02A0 424 10$:
0000 02A0 425 .WORD 0 ; entry mask
0549'CF 00 DD 02A2 426 PUSHL #0 ; push a dummy parameter
0001 01 FB 02A4 427 CALLS #1,W^REG_SAVE ; save a reg snapshot
02A9 428 $CANEXH S W^EXEC ; cancel exec exit handler #1
02B4 429 FAIL_CHECKNP SSS NORMAL ; check for success
02B4 429 PUSHL #SS$ NORMAL
005EA'CF 01 DD 02B6 430 CALLS #1,W^REG_CHECKNP
0089'CF 01CF'CF DE 02BB 431 MOVAL W^EXEC3,W^CAN+CANEXH$_DESBK ; set handler adr
02C2 431 $CANEXH G W^CAN ; cancel exec exit handler #3
02CB 432 FAIL_CHECKNP SSS NORMAL ; check for success
02CB 432 PUSHL #SS$ NORMAL
05EA'CF 01 DD 02CD 433 RET ; return
0004 04 02D2 434 20$:
0663'CF 00 FB 02D3 435 CALLS #0,W^ERLBUF_DUMP ; dump any errors that occurred
02D8 436 :+
02D8 437 :
02D8 438 : test super mode exit handler cancellation
02D8 439 :
02D8 440 :-
02D8 441 NEXT_TEST
02D8
02D8 STP10:
0004'CF 0A DO 02D8 MOVL #10,W^CURRENT_TC
0000 00 DD 02DD PUSHL #0
0549'CF 01 FB 02DF CALLS #1,W^REG_SAVE
0069'CF 012F'CF DE 02E4 442 MOVAL W^SM,W^MODE ; set the mode
0005 05 BE 02EB 443 CHMS #5 ; cancel super exit handlers #1 and #3
```



```
02ED 445 ;+
02ED 446 ;
02ED 447 ; test user mode exit handler cancellation
02ED 448 ;
02ED 449 ; -
02ED 450 NEXT_TEST
02ED
02ED STP11:
0004'CF 0B DD 02ED
0549'CF 01 DD 02F2
0069'CF 0123'CF DE 02F4
0553'CF 01 DD 02F9 451
0089'CF 016B'CF DE 0300 452
0553'CF 01 DD 030B 453
0553'CF 01 DD 030B
0553'CF 01 DD 030D
0553'CF 01 DD 0312 454
0553'CF 01 DD 0319 455
0553'CF 01 DD 0322 456
0553'CF 01 DD 0322
0553'CF 01 DD 0324

MOVL #11,W^CURRENT_TC
PUSHL #0
CALLS #1,W^REG_SAVE
MOVAL W^UM,W^MODE ; set the mode
SCANEXH S W^USER1 ; cancel user exit handler #1
FAIL_CHECK $$$_NORMAL ; check for success
PUSHL #$$$_NORMAL
CALLS #1,W^REG_CHECK
MOVAL W^USER3,W^CAN+CANEXHS_DESBLK ; set handler adr
SCANEXH G W^CAN ; cancel user exit handler #3
FAIL_CHECK $$$_NORMAL ; check for success
PUSHL #$$$_NORMAL
CALLS #1,W^REG_CHECK
```



```
0329 458 .SBTTL DCLEXH TESTS #2
0329 459 :+
0329 460 :
0329 461 : $DCLEXH tests
0329 462 :
0329 463 : This is the second of two parts of the DCLEXH tests.
0329 464 : This part tests the servicing of the exit handlers.
0329 465 : At this time there should be 2 user mode exit handlers declared.
0329 466 :
0329 467 : test user mode exit handler #4
0329 468 :
0329 469 :-
0329 470 NEXT_TEST
0329
0329 STP12:
0329      MOVL #12,W^CURRENT_TC
0329      PUSHL #0
0329      CALLS #1,W^REG_SAVE
0329      MOVAL W^DCLEXH,W^SERV_NAME ; set service name
0329      MOVL S^SS$ NORMAL,W^STATUS ; set the expected status return
0329      $EXIT_S W^MOD_MSG_CODE ; kick off ALL exit handlers
0329
0329 USERH4:
0329      .WORD 0
0329      MOVZBL S^#4,R2 ; set expected handler code
0329      BRB HNDLR_COM
0329
0329 :+
0329 478 : test user exit handler #2
0329 479 :
0329 480 :
0329 481 :-
0329 482 USERH2:
0329 483      .WORD 0
0329 484      NEXT_TEST
0329 485
0329
0329 STP13:
0329      MOVL #13,W^CURRENT_TC
0329      PUSHL #0
0329      CALLS #1,W^REG_SAVE
0329      MOVZBL S^#2,R2 ; set expected handler code
0329      HNDLR_COM:
0329      CMPL @B^4(AP),W^STATUS ; is the status adr field OK?
0329      BEQL 10$ ; br if yes
0329      PUSHL 4(AP) ; push received code
0329      PUSHAL W^STATUS ; push expected code
0329      PUSHAL W^EXP ; push string variable
0329      CALLS #3,W^PRINT_FAIL ; print the error
0329      CALLS #0,W^MODE_ID ; identify the handler mode
0329
0329 10$:
0329      CMPL R2,8(AP) ; is the argument field OK?
0329      BEQL 20$ ; br if yes
0329      PUSHL 8(AP) ; push received code
0329      PUSHL R2 ; push expected code
0329      PUSHAL W^EXP ; push string variable
0329      CALLS #3,W^PRINT_FAIL ; print the error
0329      CALLS #0,W^MODE_ID ; identify the exit handler mode
0329
0329 20$:
0329      CMPB S^#2,8(AP) ; is this the last handler?
```

0004'CF 0C DO 0329  
0549'CF 01 FB 032E  
0103'CF 0038'CF DE 0330  
0065'CF 01 DO 0335  
0341 471  
034C 472  
034C 473  
52 04 0000 034C 474  
11 11 9A 034E 475  
0351 476  
0353 477  
0353 478  
0353 479  
0353 480  
0353 481  
0353 482  
0000 0353 483  
0355 484  
0355 485  
0355  
0004'CF 0D DO 0355  
0549'CF 01 FB 035A  
52 02 9A 035C  
0065'CF 04 BC D1 0361  
15 13 0364  
04 AC DD 036A  
0065'CF DF 036C  
014D'CF DF 036F  
0691'CF 03 FB 0373  
0793'CF 00 FB 0377  
08 AC 52 D1 0381  
13 13 0385  
08 AC DD 0387  
52 DD 038A  
014D'CF DF 038C  
0691'CF 03 FB 0390  
0793'CF 00 FB 0395  
08 AC 02 91 039A



```
01 13 039E 505 BEQL 30$ ; br if yes
04 03A0 506 RET ; do the next handler
03A1 507 30$:
00 DD 03A1 508 PUSHL #0 ; push dummy parameter
0069'CF 01 FB 03A3 509 CALLS #1,W^REG_SAVE ; save the registers
012F'CF DE 03A8 510 MOVAL W^SM,W^MODE ; set the mode
03 BE 03AF 511 CHMS #3 ; reset the CHMS handler
0133'CF 0044'CF D0 03B1 512 MOVL W^MOD_MSG_CODE,W^MSGVEC1+4 ; set message code
013B'CF 0048'CF D0 03B8 513 MOVL W^TMN_ADDR,W^MSGVEC1+12 ; set up parameters
013F'CF 004C'CF D0 03BF 514 MOVL W^TMD_ADDR,W^MSGVEC1+16
03C6 515 $PUTMSG_S W^MSGVEC1 ; print the message
1C 01 F0 03D7 516 INSV #1,#STSSV_INHIB_MSG,- ; set inhibit printing on the exit status
00000044'EF 01 D0 03DA 517 #1,MOD_MSG_CODE ; save the new code in R0
50 0044'CF D0 03E0 518 MOVL W^MOD_MSG_CODE,R0
04 03E5 519 RET ; leave for good!
```



```
03E6 522 .SBTTL SUPER_MODE
03E6 523 :++
03E6 524 : FUNCTIONAL DESCRIPTION:
03E6 525 : Routine to handle the CHMS instructions.
03E6 526 :
03E6 527 : CALLING SEQUENCE:
03E6 528 : CHMS #N
03E6 529 :
03E6 530 : INPUT PARAMETERS:
03E6 531 : SP=> CHMS parameter
03E6 532 : PC
03E6 533 : PSL
03E6 534 :
03E6 535 : The CHMS parameter can be one of the following:
03E6 536 :
03E6 537 : 1 = execute a $DCLCHM_G to declare a dummy handler
03E6 538 : 2 = execute a $DCLCMH_G to clear the dummy CHMS handler
03E6 539 : 3 = execute a $DCLCMH_S to reset the CHMS handler
03E6 540 : 4 = declare 2 exit handlers in super mode
03E6 541 : 5 = delete 2 exit handlers in super mode
03E6 542 :
03E6 543 : OUTPUT PARAMETERS:
03E6 544 : NONE
03E6 545 : --
03E6 546 :
03E6 547 SUPER_MODE:
05 50 8E DO 03E6 548 MOVL (SP)+,R0 ; get CHM parameter off the stack
01 01 50 8F 03E9 549 CASEB R0,#1,#5 ; do the right thing
03ED 550 10$:
000A' 03ED 551 .WORD 20$-10$
001D' 03EF 552 .WORD 30$-10$
0037' 03F1 553 .WORD 40$-10$
0058' 03F3 554 .WORD 50$-10$
0083' 03F5 555 .WORD 60$-10$
03F7 556 20$:
03F7 557 $DCLCMH G W^DCL ; declare a dummy CHMS handler
0400 558 FAIL_CHECK SS$ _NORMAL ; check for success
0553'CF 01 DD 0400 PUSHL #SS$ _NORMAL
01 01 FB 0402 CALLS #1,W^REG_CHECK
008F 31 0407 559 BRW 70$ ; carry on
040A 560 30$:
0071'CF FFD8 CF DE 040A 561 MOVAL W^SUPER_MODE,W^DCL+DCLCMH$_ADDRS ; set up to delete dummy handler
0411 562 $DCLCMH G W^DCL ; clear the dummy handler
041A 563 FAIL_CHECK SS$ _NORMAL ; check for success
0553'CF 01 DD 041A PUSHL #SS$ _NORMAL
01 01 FB 041C CALLS #1,W^REG_CHECK
0075 31 0421 564 BRW 70$ ; carry on
0424 565 40$:
0103'CF 0031'CF DE 0424 566 MOVAL W^DCLCMH,W^SERV_NAME ; set service name pointer
042B 567 $DCLCMH S @PRVHND1,#0 ; reset the CHMS handler for DCL
043C 568 FAIL_CHECK SS$ _NORMAL ; check for success
0553'CF 01 DD 043C PUSHL #SS$ _NORMAL
01 01 FB 043E CALLS #1,W^REG_CHECK
54 11 0443 569 BRB 70$ ; carry on
0445 570 50$:
0445 571 $DCLEXH S W^SUPER1 ; declare #1 super mode exit handler
0450 572 FAIL_CHECK SS$ _NORMAL ; check for success
```



```
0081'CF 01 DD 0450          PUSHL #SS$ NORMAL
0553'CF 01 FB 0452          CALLS #1,W^REG_CHECK
01A7'CF DE 0457 573        MOVAL W^SUPER3,W^DCL1+DCLEXH$_DESBK ; set handler adr for #3
045E 574          $DCLEXH G W^DCL1 ; declare #3 super mode exit handler
0467 575          FAIL_CHECK SS$ _NORMAL ; check for success
0553'CF 01 DD 0467          PUSHL #SS$ NORMAL
01 FB 0469          CALLS #1,W^REG_CHECK
29 11 046E 576        BRB 70$ ; carry on
0470 577 60$:          $CANEXH S W^SUPER1 ; delete #1 super mode exit handler
0470 578          FAIL_CHECK SS$ _NORMAL ; check for success
047B 579
0553'CF 01 DD 047B          PUSHL #SS$ NORMAL
01 FB 047D          CALLS #1,W^REG_CHECK
0089'CF 01A7'CF DE 0482 580        MOVAL W^SUPER3,W^CAN+CANEXH$_DESBK ; set handler adr for #3
0489 581          $CANEXH G W^CAN ; delete #3 super mode exit handler
0492 582          FAIL_CHECK SS$ _NORMAL ; check for success
0553'CF 01 DD 0492          PUSHL #SS$ NORMAL
01 FB 0494          CALLS #1,W^REG_CHECK
0499 583 70$:          REI ; go back to user mode
02 0499 584          DUMMY:
049A 585          BRW SUPER_MODE ; dummy handler address
FF49 31 049A 586          .SBTTL USER_MODE
049D 587          ;++
049D 588          : FUNCTIONAL DESCRIPTION:
049D 589          : Routine to handle the CHMU instruction
049D 590          :
049D 591          : CALLING SEQUENCE:
049D 592          : CHMU #5
049D 593          :
049D 594          : INPUT PARAMETERS:
049D 595          : SP=> #5
049D 596          : PC
049D 597          : PSL
049D 598          :
049D 599          : OUTPUT PARAMETERS:
049D 600          : NONE
049D 601          :
049D 602          :--
049D 603          :
049D 604          :
049D 605          USER_MODE:
50 8E D0 049D 606          MOVL (SP)+,R0 ; get CHM parameter off the stack
05 50 D1 04A0 607          CMPL R0,S^#5 ; is it correct?
0D 13 04A3 608          BEQL 10$ ; br if yes
50 DD 04A5 609          PUSHL R0 ; save received
05 DD 04A7 610          PUSHL S^#5 ; save expected
C14D'CF 05 DD 04A9 611          PUSHAL W^EXP ; save the string variable
0691'CF 03 FB 04AD 612          CALLS #3,W^PRINT_FAIL ; print the error message
04B2 613 10$:          REI ; return
02 04B2 614
```



```
04B3 616 .SBTTL COMP_MODE
04B3 617 :++
04B3 618 : FUNCTIONAL DESCRIPTION:
04B3 619 : Compatibility mode exception handler
04B3 620 :
04B3 621 : CALLING SEQUENCE:
04B3 622 : execute a compatibility mode exception
04B3 623 :
04B3 624 : INPUT PARAMETERS:
04B3 625 : NONE
04B3 626 :
04B3 627 : OUTPUT PARAMETERS:
04B3 628 : NONE
04B3 629 :
04B3 630 :--
04B3 631 COMP_MODE:
FC A0 95 04B3 632 TSTB -4(R0) ; see if we got the correct exception
OE 13 04B6 633 BEQL 10$ ; br if correct
FC A0 DD 04B8 634 PUSHL -4(R0) ; push received code
00 DD 04BB 635 PUSHL #0 ; push expected code
014D'CF DF 04BD 636 PUSHAL W^EXP ; push string variable
0691'CF 03 FB 04C1 637 CALLS #3,W^PRINT_FAIL ; print the error
04C6 638 10$:
FFFF 8F FC8C CF B1 04C6 639 CMPW W^TEST,#-1 ; were we really in compatibility mode?
14 13 04CD 640 BEQL 20$ ; br if yes
7E FC83 CF 3C 04CF 641 MOVZWL W^TEST,-(SP) ; push received code
0000FFFF 8F DD 04D4 642 PUSHL #^X0000FFFF ; push expected code
014D'CF DF 04DA 643 PUSHAL W^EXP ; push string variable
0691'CF 03 FB 04DE 644 CALLS #3,W^PRINT_FAIL ; print the error
04E3 645 20$:
FC72 31 04E3 646 BRW RETURN ; carry on
```



```
04E6 648 .SBTTL SETUP_SUPER ROUTINE
04E6 649 :++
04E6 650 : FUNCTIONAL DESCRIPTION:
04E6 651 : Routine to declare an initial CHMS handler from user mode.
04E6 652 :
04E6 653 : CALLING SEQUENCE:
04E6 654 : $CMKRNL_S W^SETUP_SUPER,ARGLST
04E6 655 :
04E6 656 : ARGLST = address of a pointer to a one parameter argument list conta
04E6 657 : the address of the entry mask of the CHMS handler
04E6 658 :
04E6 659 : INPUT PARAMETERS:
04E6 660 : ARGLST
04E6 661 :
04E6 662 : IMPLICIT INPUTS
04E6 663 : NONE
04E6 664 :
04E6 665 : OUTPUT PARAMETERS:
04E6 666 : Declares a change mode handler for super mode which must be
04E6 667 : reset to DCL in the users handler routine when the handler is
04E6 668 : no longer needed.
04E6 669 :
04E6 670 : IMPLICIT OUTPUTS:
04E6 671 : NONE
04E6 672 :
04E6 673 : COMPLETION CODES:
04E6 674 : NONE
04E6 675 :
04E6 676 : SIDE EFFECTS:
04E6 677 : NONE
04E6 678 :
04E6 679 : ON ENTRY:
04E6 680 :
04E6 681 :
04E6 682 :
04E6 683 :
04E6 684 :
04E6 685 :
04E6 686 :
04E6 687 :
04E6 688 :
04E6 689 :
04E6 690 :
04E6 691 :
04E6 692 :
04E6 693 :
04E6 694 :--
```

KSP =>

0
0
AP
FP
PC
0
0
AP
FP
SRVEXIT
PC
PSL

USP =>

USER
CALL
FRAME



```
00000000 04E6 696 RETURN_PC:
04E6 697 .LONG 0 ; storage for user return PC
00000000 04EA 698 HANDLER_PC:
04EA 699 .LONG 0 ; storage for handler PC
04EE 700 ;
04EE 701 SETUP_SUPER:
04EE 702 .WORD ^M<R2,R3>
04F0 703 MFPR #PRS_USP,R3 ; get the user call frame address
EE AF 53 03 DB 04F3 704 MOVL SF$SAVE_PC(R3),B^RETURN_PC ; get the user return PC
ED AF 10 A3 D0 04F8 705 MOVL 4(APT,HANDLER_PC) ; save the handler address
52 04 AC D0 04FD 706 MOVL SF$SAVE_FP(FP),R2 ; get saved FP
52 0C AD D0 0501 707 ADDL S^#EXESC CMSTKSZ,R2 ; back over change mode stack frame
62 12 AF 9E 0504 708 MOVAB B^20$(R2) ; set return address
0A FO 0508 709 INSV #<<PSL$C_SUPER@PSL$S_CURMOD>+PSL$C_SUPER>,-
050A 710 #PSL$V_PRVMOD,-
04 A2 04 050B 711 #PSL$S_CURMOD*2,4(R2) ; set current and previous mode to super
50 01 D0 050E 712 MOVL S^#SS$_NORMAL,R0 ; set correct return code
04 0511 713 RET ; enter super mode
0512 714 20$:
18 AF 7E D4 0512 715 CLRL -(SP) ; set up dummy PSL
6E FA 0514 716 CALLG (SP),B^30$ ; create initial call frame
0518 717 30$:
0000 0518 718 .WORD ^M<> ; entry mask
051A 719 PUSHL #0 ; push a dummy parameter
0549 CF 01 FB 051C 720 CALLS #1,W^REG_SAVE ; save the registers
0069 CF 012F CF DE 0521 721 MOVAL W^SM,W^MODE ; set the mode
0528 722 $DCLCMH S @HANDLER_PC,W^PRVHND1,#0 ; set real handler
0538 723 FAIL_CHECKNP SS$ NORMAL ; check for success
0538 724 PUSHL #SS$ NORMAL
05EA CF 01 FB 053A 725 CALLS #1,W^REG_CHECKNP
03C00000 8F DD 053F 726 PUSHL #<<PSL$C_USER@PSL$V_CURMOD>-
9E AF DD 0545 727 !<PSL$C_USER@PSL$V_PRVMOD>>; set return to user
02 0548 728 PUSHL RETURN_PC ; set the return PC
0549 729 .SBTTL REG_SAVE ; return to user mode
0549 729 :++
0549 730 : FUNCTIONAL DESCRIPTION:
0549 731 : Subroutine to save R2-R11 in the register save location.
0549 732 :
0549 733 : CALLING SEQUENCE:
0549 734 : PUSHL #0 ; save a dummy parameter
0549 735 : CALLS #1,W^REG_SAVE ; save R2-R11
0549 736 :
0549 737 : INPUT PARAMETERS:
0549 738 : NONE
0549 739 :
0549 740 : OUTPUT PARAMETERS:
0549 741 : NONE
0549 742 :
0549 743 :--
0549 744 :
0008 CF 14 AD 28 OFFC 0549 745 REG_SAVE:
0549 746 .WORD ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
054B 747 MOVBC3 #4*10,^X14(FP),W^REG_SAVE_AREA ; save the registers in the program
0552 748 RET
```



```
0553 750 .SBTTL REG_CHECK
0553 751 :++
0553 752 : FUNCTIONAL DESCRIPTION:
0553 753 : Subroutine to test R0 & R2-R11 for proper content after a service
0553 754 : execution. A snapshot is taken by the REG_SAVE routine at the
0553 755 : beginning of each step and this routine is executed after the
0553 756 : services have been executed.
0553 757 :
0553 758 : CALLING SEQUENCE:
0553 759 : PUSHL #SS$ XXXXXX ; push expected R0 contents
0553 760 : CALLS #1,W^REG_CHECK ; execute this routine
0553 761 :
0553 762 : INPUT PARAMETERS:
0553 763 : expected R0 contents on the stack
0553 764 :
0553 765 : OUTPUT PARAMETERS:
0553 766 : possible error messages printed using $PUTMSG
0553 767 :
0553 768 :--
0553 769 :
0553 770 REG_CHECK:
0553 771 .WORD ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
16 50 04 AC D1 0555 772 CMPL 4(AP),R0 ; is this the right fail code?
0553 773 BEQL 10$ ; br if yes
0553 774 PUSHL R0 ; push received data
0553 775 PUSHL 4(AP) ; push expected data
0553 776 PUSHAL W^EXP ; push the string variable
0553 777 CALLS #3,W^PRINT_FAIL ; print the error message
0553 778 10$:
0553 779 CMPC3 #4*10,^X14(FP),W^REG_SAVE_AREA ; check all but R0
0553 780 BEQL 20$ ; br if O.K.
56 53 00000008'8F C3 0572 781 SUBL3 #REG_SAVE_AREA,R3,R6 ; calculate the register number
0553 782 DIVL2 #4,R6
0553 783 ADDB3 #^X2,R6,-(SP) ; set number past R0-R1 and save
0553 784 BICL2 #3,R1 ; backup to register boundrys
0553 785 BICL2 #3,R3
0553 786 PUSHL (R1) ; push received data
0553 787 PUSHL (R3) ; push expected data
0553 788 PUSHAL W^REG ; set string ptr param.
0553 789 CALLS #4,W^PRINT_FAIL ; print the error message
0553 790 20$:
0553 791 RET
```

0691'CF 03 FB 0564 777  
0008'CF 14 AD 28 29 0569 779  
56 53 00000008'8F C3 0572 781  
7E 56 04 C6 057A 782  
51 03 CA 0581 784  
53 03 CA 0584 785  
61 DD 0587 786  
63 DD 0589 787  
008D'CF DF 058B 788  
0691'CF 04 FB 058F 789  
04 0594 790  
04 0594 791



```
0595 793 .SBTTL REG_CHECKNP
0595 794 :++
0595 795 : FUNCTIONAL DESCRIPTION:
0595 796 : Subroutine to test R0 & R2-R11 for proper content after a service
0595 797 : execution without printing it. A snapshot is taken by the REG_SAVE routine a
0595 798 : beginning of each step and this routine is executed after the
0595 799 : services have been executed. This routine collects the error
0595 800 : information in buffer ERLB instead of printing it.
0595 801 :
0595 802 : CALLING SEQUENCE:
0595 803 : PUSHL #SS$ XXXXXX ; push expected R0 contents
0595 804 : CALLS #1,W*REG_CHECK ; execute this routine
0595 805 :
0595 806 : INPUT PARAMETERS:
0595 807 : expected R0 contents on the stack
0595 808 :
0595 809 : OUTPUT PARAMETERS:
0595 810 : possible error messages logged in buffer ERLB which are printed
0595 811 : using routine ERLBUF_DUMP.
0595 812 :
0595 813 :--
0595 814 :
0595 815 FLAG:
00 0595 816 .BYTE 0 ; error flags are BIT0 = 0 means no errors in the bu
0596 817 ; BIT0 = 1 means errors in the buffe
0596 818 ELBP:
0000059A' 0596 819 .ADDRESS ERLB ; error log buffer pointer
059A 820 ERLB:
000005EA 059A 821 .BLKB 80 ; error log buffer
05EA 822 :
05EA 823 REG_CHECKNP:
05EA 824 .WORD ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
50 04 AC D1 05EC 825 CMPL 4(AP),R0 ; is this the right fail code
2B 13 05F0 826 BEQL 10$ ; br if yes
9F AF 01 88 05F2 827 BISB2 #1,FLAG ; set the error logged flag bit
52 9D AF D0 05F6 828 MOVL ELBP,R2 ; get the current error log pointer
82 03 90 05FA 829 MOVB #3,(R2)+ ; save the long word count
82 50 D0 05FD 830 MOVL R0,(R2)+ ; save received status
82 04 AC D0 0600 831 MOVL 4(AP),(R2)+ ; save expected status
82 014D'CF DE 0604 832 MOVAL W^EXP,(R2)+ ; save the string variable
87 AF 52 D0 0609 833 CLRL (R2) ; set the terminator
004C'CF 002A'CF DE 060B 834 MOVL R2,ELBP ; reset the buffer pointer
0044'CF 03 00 02 F0 060F 835 MOVAL W^TEST_MOD_FAIL,W^TMD_ADDR ; set failure message address
0008'CF 14 AD 28 29 061D 836 INSV #ERROR,#0,#3,W^MOD_MSG_CODE ; set severity code
061D 837 10$:
061D 838 CMPC3 #4*10,^X14(FP),W^REG_SAVE_AREA ; check all but R0 and R1
FF6A CF 01 88 0624 839 BEQL 20$ ; br if OK
52 FF67 CF D0 0626 840 BISB2 #1,FLAG ; set error logged flag bit
82 04 90 062B 841 MOVL ELBP,R2 ; get current error log buf pointer
00000008'8F C3 0630 842 MOVB S^#4,(R2)+ ; set longword count
56 53 0633 843 SUBL3 #REG_SAVE_AREA,-
82 56 04 C6 0639 844 R3,R6 ; calc reg number
56 02 C1 063B 845 DIVL2 S^#4,R6 ; make it a longword count
82 82 61 D0 063E 846 ADDL3 S^#2,R6,(R2)+ ; correct for R0-R1 and save
82 82 63 D0 0642 847 MOVL (R1),(R2)+ ; save received results
82 008D'CF DE 0645 848 MOVL (R3),(R2)+ ; save expected results
0648 849 MOVAL W^REG,(R2)+ ; save string variable
```



SATSSS43  
V04-000

- SATS SYSTEM SERVICE TESTS (SUCC S.C.) 16-SEP-1984 00:54:19 VAX/VMS Macro V04-00 Page 25  
REG\_CHECKNP 5-SEP-1984 04:31:29 [UETPSY.SRC]SATSSS43.MAR;1 (2)

0044'CF	FF42	CF	62	D4	064D	850	CLRL	(R2)	; set the terminator
			52	D0	064F	851	MOVL	R2,ELBP	; reset the buffer pointer
	004C'CF	002A'CF		DE	0654	852	MOVAL	W^TEST_MOD_FAIL,W^TMD_ADDR	; set failure message address
	03	00	02	F0	065B	853	INSV	#ERROR,#0,#3,W^MOD_MSG_CODE	; set severity code
					0662	854			
				04	0662	855	RET		; bail out



```
0663 857 .SBTTL ERLBUF_DUMP
0663 858 :++
0663 859 : FUNCTIONAL DESCRIPTION:
0663 860 : Routine to check for errors in the error log buffer and
0663 861 : report any that are there.
0663 862 :
0663 863 : CALLING SEQUENCE:
0663 864 : CALLS #0,W^ERLBUF_DUMP
0663 865 :
0663 866 : INPUT PARAMETERS:
0663 867 : FLAG bit 0 = 0 for no errors logged
0663 868 : FLAG bit 0 = 1 for errors logged
0663 869 : if errors logged then buffer ERLB must contain legal format errors
0663 870 :
0663 871 : OUTPUT PARAMETERS:
0663 872 : NONE
0663 873 :
0663 874 :--
0663 875 :
0663 876 ERLBUF_DUMP:
0663 877 .WORD ^M<R2,R3,R4>
1B FF2C CF 001C 0665 878 BLBC FLAG,30$ ; br if no errors to report
52 FF2C CF DE 066A 879 MOVAL ERLB,R2 ; set up buffer pointer
066F 880 10$:
62 D5 066F 881 TSTL (R2) ; any more errors?
12 13 0671 882 BEQL 30$ ; br if not
53 82 9A 0673 883 MOVZBL (R2)+,R3 ; get the longword count
54 53 D0 0676 884 MOVL R3,R4 ; and save it
0679 885 20$:
82 DD 0679 886 PUSHL (R2)+ ; push a parameter
FB 53 F5 067B 887 SOBGTR R3,20$ ; and push them all
0691'CF 54 FB 067E 888 CALLS R4,W^PRINT_FAIL ; print the failure
EA 11 0683 889 BRB 10$ ; do the next one
0685 890 30$:
FFOA CF FF11 CF DE 0685 891 MOVAL ERLB,ELBP ; reset the buffer pointer
FFOA CF D4 068C 892 CLRL W^ERLB ; set fresh terminator
04 0690 893 RET ; bail out
```



```
0691 895 .SBTTL PRINT_FAIL
0691 896 :++
0691 897 : FUNCTIONAL DESCRIPTION:
0691 898 : Subroutine to report failures using $PUTMSG
0691 899 :
0691 900 : CALLING SEQUENCE:
0691 901 : Mode #1      PUSHL EXPECTED Mode #2      PUSHL REG_NUMBER
0691 902 :              PUSHL RECEIVED              PUSHL EXPECTED
0691 903 :              PUSHAL STRING_VAR            PUSHL RECEIVED
0691 904 :              CALLS #3,W^PRINT_FAIL        PUSHAL STRING_VAR
0691 905 :                                           CALLS #4,W^PRINT_FAIL
0691 906 : INPUT PARAMETERS:
0691 907 : listed above
0691 908 :
0691 909 : OUTPUT PARAMETERS:
0691 910 : an error message is printed using $PUTMSG
0691 911 :
0691 912 :--
0691 913
0691 914 PRINT_FAIL:
003C 0691 915 .WORD ^M<R2,R3,R4,R5>
0693 916 $FAO S W^CS1,W^MESSAGEL,W^MSGL,#TEST_MOD_NAME,W^SERV_NAME,W^CURRENT_TC
06B4 917 $PUTMSG S W^MSGVEC ; print the message
04 6C 91 918 CMPB (AP),#4 ; is this a register message?
21 13 919 BEQL 10$ ; br if yes
25 11 920 $FAO S W^CS2,W^MESSAGEL,W^MSGL,4(AP),8(AP),4(AP),12(AP)
06E9 921 BRB 20$ ; goto output message
06EB 922 10$:
06EB 923 $FAO S W^CS3,W^MESSAGEL,W^MSGL,4(AP),16(AP),8(AP),4(AP),16(AP),12(AP)
0710 924 20$:
0710 925
0710 926 $PUTMSG S W^MSGVEC ; print the message
0721 927 CALLS #0,W^MODE_ID ; identify the mode
0726 928 MOVAL W^TEST_MOD_FAIL,W^TMD_ADDR ; set failure message address
0044'CF 03 00 02 072D 929 INSV #ERROR,#0,#3,W^MOD_MSG_CODE ; set severity code
04 0734 930 RET
0735 931 USERH1:
0735 932 USERH3:
0069'CF 0123'CF 0000 0735 933 .WORD 0
DE 14 11 0737 934 MOVAL W^UM,W^MODE ; set the mode string
073E 935 BRB CEP
0740 936 SUPERH1:
0740 937 SUPERH3:
0069'CF 012F'CF 0000 0740 938 .WORD 0
DE 09 11 0742 939 MOVAL W^SM,W^MODE ; set the mode string
0749 940 BRB CEP
074B 941 EXECH1:
074B 942 EXECH3:
0069'CF 013C'CF 0000 074B 943 .WORD 0
DE 074D 944 MOVAL W^EM,W^MODE ; set the mode string
0754 945 CEP:
0754 946 $FAO S W^CS4,W^MESSAGEL,W^MSGL,MODE,#TEST_MOD_NAME ; format the error strin
0773 947 $PUTMSG S W^MSGVEC ; print the message
0044'CF 03 00 02 0784 948 MOVAL W^TEST_MOD_FAIL,W^TMD_ADDR ; set failure message address
078B 949 INSV #ERROR,#0,#3,W^MOD_MSG_CODE ; set severity code
04 0792 950 RET
```



```

0793 952 .SBTTL MODE_ID
0793 953 :++
0793 954 : FUNCTIONAL DESCRIPTION:
0793 955 : Subroutine to identify the mode that an exit handler is in.
0793 956 :
0793 957 : CALLING SEQUENCE:
0793 958 : CALLS #0,W^MODE_ID
0793 959 :
0793 960 : INPUT PARAMETERS:
0793 961 : MODE contains an address pointing to an ascii string desc.
0793 962 : of the current CPU mode.
0793 963 :
0793 964 : OUTPUT PARAMETERS:
0793 965 : NONE
0793 966 :
0793 967 :--
0793 968 :
0793 969 MODE_ID:
003C 0793 970 .WORD ^M<R2,R3,R4,R5>
0795 971 $FAO S W^CS5,W^MESSAGEL,W^MSGL,MODE ; format the error message
07AE 972 $PUTMSG_S W^MSGVEC ; print the mode message
04 07BF 973 RET

```



```
07C0 975 MOD_MSG_PRINT:
07C0 976 :
07C0 977 : *****
07C0 978 : *
07C0 979 : * PRINTS THE TEST MODULE BEGUN/SUCCESSFUL/FAILED MESSAGES *
07C0 980 : * (USING THE PUTMSG MACRO). *
07C0 981 : *
07C0 982 : *****
07C0 983 :
05 07C0 984 PUTMSG <MOD_MSG_CODE,#2,TMN_ADDR,TMD_ADDR> : PRINT MSG
07DB 985 RSB ; ... AND RETURN TO CALLER
07DC 986 :
07DC 987 CHMRTN:
07DC 988 : *****
07DC 989 : *
07DC 990 : * CHANGE MODE ROUTINE. THIS ROUTINE GETS CONTROL WHENEVER *
07DC 991 : * A CMKRNL, CMEXEC, OR CMSUP SYSTEM SERVICE IS ISSUED *
07DC 992 : * BY THE MODE MACRO ('TO' OPTION). IT MERELY DOES *
07DC 993 : * A JUMP INDIRECT ON A FIELD SET UP BY MODE. IT HAS *
07DC 994 : * THE EFFECT OF RETURNING TO THE END OF THE MODE *
07DC 995 : * MACRO EXPANSION. *
07DC 996 : *
07DC 997 : *****
07DC 998 :
00000059'FF 0000 07DC 999 .WORD 0 ; ENTRY MASK
17 07DE 1000 JMP @CHM_CONT ; RETURN TO MODE MACRO IN NEW MODE
07E4 1001 :
07E4 1002 : * RET INSTR WILL BE ISSUED IN EXPANSION OF 'MODE FROM, ....' MACRO
07E4 1003 :
07E4 1004 .END SATSSS43
```



SATSSS43  
Symbol table

- SATS SYSTEM SERVICE TESTS (SUCC S.C.) 16-SEP-1984 00:54:19 VAX/VMS Macro V04-00  
5-SEP-1984 04:31:29 [UETPSY.SRC] SATSSS43.MAR;1

Page 30  
(2)

\$\$ARGS	= 00000001		PRIVMASK	00000051 R	03
\$\$T1	= 00000004		PRVHND1	00000107 R	03
\$\$T2	= 00000004		PRVHND2	0000010B R	03
ARGLST	00000117 R	03	PRVHND3	0000010F R	03
BUF	000000AB R	03	PRVPRT	00000050 R	03
CAN	00000085 R	03	PSL\$C_SUPER	= 00000002	
CANEXH	0000003F R	02	PSL\$C_USER	= 00000003	
CANEXH\$-DESBK	= 00000004		PSL\$M_CM	= 80000000	
CANEXH\$-NARGS	= 00000001		PSL\$S_CURMOD	= 00000002	
CEP	00000754 R	04	PSL\$V_CURMOD	= 00000018	
CHMRTN	000007DC R	04	PSL\$V-PRVMOD	= 00000016	
CHM_CONT	00000059 R	03	REG	0000008D R	03
COMP_MODE	000004B3 R	04	REGNUM	0000009F R	03
CS1	00000046 R	02	REG_CHECK	00000553 R	04
CS2	00000078 R	02	REG_CHECKNP	000005EA R	04
CS3	000000A5 R	02	REG_SAVE	00000549 R	04
CS4	000000D8 R	02	REG_SAVE_AREA	00000008 R	03
CS5	0000010E R	02	RETADR	0000005D R	03
CURRENT_TC	00000004 R	03	RETURN	00000158 R	04
DCL	0000006D R	03	RETURN_PC	000004E6 R	04
DCL1	0000007D R	03	SATSSS43	00000000 RG	04
DCLCMH	00000031 R	02	SERV_NAME	00000103 R	03
DCLCMH\$-ADDRES	= 00000004		SETUP_SUPER	000004EE R	04
DCLCMH\$-NARGS	= 00000003		SEVERE	= 00000004	
DCLCMH\$-PRVHND	= 00000008		SF\$L_SAVE_FP	= 0000000C	
DCLCMH\$-TYPE	= 0000000C		SF\$L_SAVE_PC	= 00000010	
DCLEXH	00000038 R	02	SHR\$K_SHRDEF	= 00000001	
DCLEXH\$-DESBK	= 00000004		SHR\$-TEXT	= 00001130	
DCLEXH\$-NARGS	= 00000001		SM	0000012F R	02
DUMMY	0000049A R	04	SS\$ NORMAL	= 00000001	
ELBP	00000596 R	04	STATUS	00000065 R	03
EM	0000013C R	02	STEP	= 0000000D	
ERLB	0000059A R	04	STP0	0000003D R	04
ERLBUF_DUMP	00000663 R	04	STP1	00000078 R	04
ERROR	= 00000002		STP10	C00002D8 R	04
EXE\$C_CMSTKSZ	***** X	04	STP11	000002ED R	04
EXEC1	000001BB R	03	STP12	00000329 R	04
EXEC3	000001CF R	03	STP13	00000355 R	04
EXECH1	0000074B R	04	STP2	000000DD R	04
EXECH3	0000074B R	04	STP3	00000138 R	04
EXP	0000014D R	02	STP4	00000158 R	04
FLAG	00000595 R	04	STP5	00000166 R	04
HANDLER_PC	000004EA R	04	STP6	000001A5 R	04
HNDLR_COM	00000364 R	04	STP7	000001FE R	04
INFO	= 00000003		STP8	00000213 R	04
LIB\$SIGNAL	***** X	04	STP9	00000278 R	04
MESSAGEL	000000FB R	03	ST\$SV INHIB_MSG	= 0000001C	
MODE	00000069 R	03	SUCCESS	= 00000001	
MODE_ID	00000793 R	04	SUPER1	00000193 R	03
MOD_MSG_CODE	00000044 R	03	SUPER3	000001A7 R	03
MOD_MSG_PRINT	000007C0 R	04	SUPERH1	00000740 R	04
MSGC	000000A3 R	03	SUPERH3	00000740 R	04
MSGVEC	0000011F R	03	SUPER_MODE	000003E6 R	04
MSGVEC1	0000012F R	03	SYSSCANEXH	***** GX	04
OHC	00000113 R	03	SYSSCMEXEC	***** GX	04
PR\$ USP	= 00000003		SYSSCMKRNL	***** GX	04
PRINT_FAIL	00000691 R	04	SYSSDCLCMH	***** GX	04



SATSSS43  
Symbol table

```
SYSSDCLEXH ***** GX 04
SYSS$EXIT ***** GX 04
SYSS$FAO ***** X 04
SYSS$HIBER ***** GX 04
SYSS$PUTMSG ***** GX 04
SYSS$SETPRN ***** GX 04
SYSS$WAKE ***** GX 04
TEST 00000156 R 04
TEST_MOD_BEGIN 00000019 R 02
TEST_MOD_FAIL 0000002A R 02
TEST_MOD_NAME 00000000 R 02
TEST_MOD_NAME_D 00000009 R 02
TEST_MOD_SUCC 0000001F R 02
TMD_ADDR 0000004C R 03
TMN_ADDR 00000048 R 03
TPID 00000000 R 03
UETPS_SATSMS = 007480D9
UETPS_TEXT = 00741133
UM 00000123 R 02
USER1 00000143 R 03
USER2 00000157 R 03
USER3 0000016B R 03
USER4 0000017F R 03
USERH1 00000735 R 04
USERH2 00000353 R 04
USERH3 00000735 R 04
USERH4 0000034C R 04
USER_MODE 0000049D R 04
WARNING = 00000000
```

+-----+  
! Psect synopsis !  
+-----+

PSECT name	Allocation	PSECT No.	Attributes
ABS	00000000 ( 0.)	00 ( 0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
\$ABSS	00000000 ( 0.)	01 ( 1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
RODATA	0000015B ( 347.)	02 ( 2.)	NOPIC USR CON REL LCL NOSHR NOEXE RD NOWRT NOVEC LONG
RWDATA	000001E3 ( 483.)	03 ( 3.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC LONG
SATSSS43	000007E4 ( 2020.)	04 ( 4.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG

+-----+  
! Performance indicators !  
+-----+

Phase	Page faults	CPU Time	Elapsed Time
Initialization	37	00:00:00.07	00:00:00.30
Command processing	133	00:00:00.66	00:00:02.62
Pass 1	411	00:00:14.67	00:00:25.19
Symbol table sort	0	00:00:01.91	00:00:02.62
Pass 2	206	00:00:03.72	00:00:08.03
Symbol table output	18	00:00:00.14	00:00:00.34
Psect synopsis output	2	00:00:00.02	00:00:00.03
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	809	00:00:21.19	00:00:39.13



The working set limit was 1800 pages.  
88714 bytes (174 pages) of virtual memory were used to buffer the intermediate code.  
There were 60 pages of symbol table space allocated to hold 1170 non-local and 32 local symbols.  
1004 source lines were read in Pass 1, producing 28 object records in Pass 2.  
52 pages of virtual memory were used to define 48 macros.

+-----+  
! Macro library statistics !  
+-----+

Macro library name	Macros defined
-----	-----
\$255\$DUA28:[SHRLIB]UETP.MLB;1	10
\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	2
\$255\$DUA28:[SYSLIB]STARLET.MLB;2	32
TOTALS (all libraries)	44

1367 GETS were required to define 44 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS\$:SATSSS43/OBJ=OBJ\$:SATSSS43 MSRC\$:SATSSS43/UPDATE=(ENH\$:SATSSS43)+EXECML\$/LIB+SHRLIB\$:UETP/LIB



0423

DIGITAL EQUIPMENT CORPORATION  
CONFIDENTIAL AND PROPRIETARY